Hw1 report

Methods in detail

Overview

在這次的作業裡，我使用的是 Spacy 的套件，共實作了三個主要的方法，其中當然還包括一些小細節，最主要的方法還是要看 token.dep_。

🏷️ SUBJECTTAG = ["nsubj", "nsubjpass", "csubj", "csubjpass", "agent", "expl"]

🏷️ OBJECTTAG = ["dobj", "pobj", "dative", "attr", "oprd"]

Method 1 - token.dep_

💡 Score: 0.6767

這個方法是我最一開始實作的，一開始其實毫無頭緒該如何處理 SVO，所以就想說直接根據token.dep_ 來判斷，結果意外地就超過 simple_baseline，以下為我找主詞、動詞、受詞的做法：

- 主詞：如果 token.dep_ 屬於 SUBJECTTAG 其中一個，就將它視為主詞。
- 動詞：如果 token.pos_ == “VERB”，就將它視為動詞。
- 受詞：如果 token.dep_ 屬於 OBJECTTAG 其中一個，就將它視為受詞。

Method 2 - subtree

💡 Score: 0.58795

原先我認為這個方法會有很好的效果，但結果似乎不如預期，後來我才發現是因為 subtree 常常會抓出很長的主詞和受詞，但這次作業的判斷是 csv 檔裡面的 S、V、O

是否有包含我找出來的 S、V、O，如果句子太長就會造成關鍵字雖然有成功找到，但因為句子過長，反而導致未包含的情況發生，而造成錯誤判斷。

- 主詞：如果 token.dep_ 屬於 SUBJECTTAG 其中一個加上 token.head 跟本次動詞的 index 相同，就將它視為主詞，而且是回傳 token.subtree。
- 動詞：如果 token.pos_ == “VERB”，就將它視為動詞。
- 受詞：如果 token.dep_ 屬於 OBJECTTAG 其中一個加上 token.head 跟本次動詞的 index 相同，就將它視為受詞，而且是回傳 token.subtree。

Method 3 - coordinated tokens

Score: 0.71473

此方法是根據方法一進行改良，因為我測試了助教提供的 example_with_answer.csv 裡的幾個句子，發現有以下問題並且處理後確實有改善：

1. 主詞應該要在動詞左邊，受詞要在動詞右邊，而且因為句子都蠻複雜的，會有好幾組 SVO，所以我認為每次其實不需要找整個句子。我想到的辦法是限定一個範圍，而不是每次都跑過一次句子，並在範圍內找主詞和受詞，範圍的大小可參考下方主詞和受詞做法。
2. 有些句子是被動語態(ex. Our house was built by my father.)，在這個句子裡，was 和 built 都不會被標記為動詞，所以會略過一組 SVO。後來我發現像這種被動語態的句子，be 動詞會被標記為 ”AUX”，所以在找動詞時，除了原本的 ”VERB”，我還會加入 ”AUX”。
3. 在較複雜的句子裡，有些主詞和動詞會被忽略，於是我便將每個主詞或受詞的 coordinated tokens 通通加進去，我認為這個部分也是方法三能有較高精準度的主要原因。

以下為我找主詞、動詞、受詞的做法：

- 主詞：有兩個參數分別為 prevID 和 verbID，負責記錄前個動詞的位置和本次動詞的位置，在找主詞的過程中，不必跑過整個句子，而是從上個動詞後面再繼續找就好，避免一直找到重複主詞，再加上主詞會在本次動詞的左邊，所以找到本次動詞即可停止，要找主詞的範圍是前個動詞到本次動詞之間。至於怎樣算主詞，幾乎都和前面一樣，如果 token.dep_ 屬於 SUBJECTTAG 其中一個，就將它視為主詞，每一輪找完該輪主詞後，還要再利用 token.conjunts 找出每個主詞的 coordinated tokens。

- 動詞：考慮到被動語態問題，只要 token.pos_ == “VERB” or “AUX”，就將它視為動詞。
- 受詞：這裡只有一個參數 verbID，負責記錄本次動詞的位置，因為受詞會在本次動詞的右邊，範圍為本次動詞到下個動詞，但因為不一定會有下個動詞，所以從本次動詞開始一路找到有動詞再結束尋找即可。至於怎樣算受詞，如果 token.dep_ 屬於 OBJECTTAG 其中一個，就將它視為受詞，每一輪找完該輪受詞後，一樣利用 token.conjunts 找出每個受詞的 coordinated tokens。

Diffrence between your expectations and the results

1. 在方法一裏頭，因為一開始對於該如何處理 SVO 沒什麼想法，所以發現 token.dep_ 會替我標示出一些主詞和受詞後，就想說直接根據這個來判斷，原本只是想說先試試看，之後再來想細節要如何處理，所以我一開始並沒有抱持多大的期待，結果意外地就超過 simple_baseline。
2. 反而是 subtree 的方式，原先我認為這個方法會有很好的效果，但結果卻不如預期，後來我發現是因為這次的作業是根據 csv 檔裡面的 S、V、O ，來看其是否有包含我找出來的 S、V、O，但是 subtree 常常會抓出很長的主詞和受詞，所以這樣的判斷方法會導致此方法雖然有成功找到主詞和受詞的關鍵字，但因為句子長度過長，而沒有達到”包含於”csv 檔裡面的 S、V、O，造成誤判。然而，有些句子裏頭的 S、V、O 可能會是片語，甚至是一個句子，這時候利用 subtree 找出來的 S、V、O 其實會更完整、更明確。
3. 方法三是唯一有符合我的預期，因為我認為除了原先該輪的主詞和受詞外，它們的 coordinated tokens 似乎是被忽略的，所以我利用了 token.conjunts 找出每個主詞和受詞的 coordinated tokens，準確率也確實提升了不少。

Difficulties and Solutions

1. 我試著從 example_with_answer.csv 的答案去找出問題點，後來發現蠻多小問題的，像是主詞前面會不小心含括介係詞、動詞分詞應該要當受詞卻被分成 ”xcomp”……等等，一開始我逐一替這些問題新增條件進行判斷，結果發現這樣處理似乎會 overfit，導致效果並不是很好。因此，後來我開始往一些大方向去思考，而不是那些比較小的問題點，就有順利提升準確率了。
2. 在比較複雜的句子結構中，一開始發現有些 token.pos_ 和 token.dep_ 標示的和我想的不太一樣，有試著要多加幾個條件，將 token.pos_ 和 token.dep_ 改成較貼近我想法的標示，但我並沒有找到真正能解決這個問題的做法，嘗試了好幾種方式，準確率提升的幅度都沒有很大。

3. 因為不知道哪些方法是可行的，所以就要一直測試，但因為一天只能交五次檔案，所以每次都要先用 example_with_answer.csv 測試過後才敢交出去。一開始如果在 example_with_answer.csv 沒有達到 65% 的準確率，我就會繼續修改，結果後來丟了一份在 example_with_answer.csv 的準確率只有 58% 的檔案到 kaggle 上，卻超過 strong baseline，超意外的！